In [1]:
```
import pandas as pd
import numpy as np
import matplotlib.pyplot as plt
```

In [2]:
```
df=pd.read_csv('Social_Network_Ads.csv')
df
```

Out[2]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0
...	...	...	...	...	...
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

400 rows × 5 columns

Explore Dataset

In [3]:
```
df.head()
```

Out[3]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	Male	19	19000	0
1	15810944	Male	35	20000	0
2	15668575	Female	26	43000	0
3	15603246	Female	27	57000	0
4	15804002	Male	19	76000	0

In [4]:
```
df.tail()
```

Out[4]:

	User ID	Gender	Age	EstimatedSalary	Purchased
395	15691863	Female	46	41000	1
396	15706071	Male	51	23000	1
397	15654296	Female	50	20000	1
398	15755018	Male	36	33000	0
399	15594041	Female	49	36000	1

In [5]:
```
df.describe
```

Out[5]:
```
<bound method NDFrame.describe of       User ID  Gender  Age  EstimatedSalary  Purchased
0    15624510    Male   19            19000          0
1    15810944    Male   35            20000          0
2    15668575  Female   26            43000          0
3    15603246  Female   27            57000          0
4    15804002    Male   19            76000          0
..        ...     ...  ...              ...        ...
395  15691863  Female   46            41000          1
396  15706071    Male   51            23000          1
397  15654296  Female   50            20000          1
398  15755018    Male   36            33000          0
399  15594041  Female   49            36000          1

[400 rows x 5 columns]>
```

In [6]:
```
df.info()
```

```
<class 'pandas.core.frame.DataFrame'>
RangeIndex: 400 entries, 0 to 399
Data columns (total 5 columns):
 #   Column           Non-Null Count  Dtype 
---  ------           --------------  ----- 
 0   User ID          400 non-null    int64 
 1   Gender           400 non-null    object
 2   Age              400 non-null    int64 
 3   EstimatedSalary  400 non-null    int64 
 4   Purchased        400 non-null    int64 
dtypes: int64(4), object(1)
memory usage: 15.8+ KB
```

In [7]:
```
df.dtypes
```

Out[7]:
```
User ID              int64
Gender              object
Age                  int64
EstimatedSalary      int64
Purchased            int64
dtype: object
```

In [8]:
```
df.isnull().sum()
```

Out[8]:
```
User ID            0
Gender             0
Age                0
EstimatedSalary    0
Purchased          0
dtype: int64
```

In [9]:
```
df.duplicated().sum()
```

Out[9]:
```
0
```

In [10]:
```
mapi={'Male':1,'Female':0}
df=df.replace(mapi)
df.head()
```

Out[10]:

	User ID	Gender	Age	EstimatedSalary	Purchased
0	15624510	1	19	19000	0
1	15810944	1	35	20000	0
2	15668575	0	26	43000	0
3	15603246	0	27	57000	0
4	15804002	1	19	76000	0

In [11]:
```
df.drop(['User ID'],axis=1, inplace=True)
df.head()
```

Out[11]:

	Gender	Age	EstimatedSalary	Purchased
0	1	19	19000	0
1	1	35	20000	0
2	0	26	43000	0
3	0	27	57000	0
4	1	19	76000	0

Train test split

In [12]:
```
x,y=df.drop(['Purchased'],axis=1),df['Purchased']
```

In [13]:
```
from sklearn.model_selection import train_test_split
xtrain,xtest,ytrain,ytest=train_test_split(x,y,test_size=0.25,random_state=0)
```

Standard scaler

In [14]:
```
from sklearn.preprocessing import StandardScaler
sc_scale=StandardScaler()

xtrain=sc_scale.fit_transform(xtrain)
xtest=sc_scale.transform(xtest)
```

In [15]:
```
from sklearn.naive_bayes import GaussianNB
classifier=GaussianNB()
classifier.fit(xtrain,ytrain)
```

Out[15]:
```
GaussianNB()
```

In [16]:
```
y_pred=classifier.predict(xtest)
```

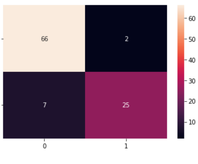
confusion matrix

In [17]:
```
from sklearn.metrics import confusion_matrix
cm=confusion_matrix(ytest,y_pred)

print("Confusion matrix is : \n",cm)
```

```
Confusion matrix is :
 [[66  2]
 [ 7 25]]
```

In [18]:
```
import seaborn as sns
import matplotlib.pyplot as plt
sns.heatmap(cm,annot=True)
plt.show()
```



Accuracy and score

In [19]:
```
from sklearn.metrics import accuracy_score
print("Accuracy :", accuracy_score(ytest,y_pred)*100,'%')
```

```
Accuracy : 91.0 %
```

In [20]:
```
from sklearn.metrics import precision_score
from sklearn.metrics import recall_score
from sklearn.metrics import f1_score
```

In [22]:
```
#precision  tp/(tp+fp)
precision=precision_score(ytest,y_pred)
print('Precision: %f' % precision)
#recall  tp/(tp+fn)
recall=recall_score(ytest,y_pred)
print('recall: %f' % recall)
#precision  2tp/(2tp+fp+fn)
f1=f1_score(ytest,y_pred)
print('f1: %f' % f1)
```

```
Precision: 0.925926
recall: 0.781250
f1: 0.847458
```

In [ ]:
```

```